Treball Fi de Màster

Màster Universitari en Enginyeria Industrial

Aplicació de les xarxes neuronals (AI) per a la presa de decisions empresarials

Autor: Gerard Colomer Castelló

Tutor: Lluís Solano Albajes

Setembre 2018

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Resum

Sembla que cada dia hi ha un descobriment o article nou relacionat amb el món de la intel·ligència artificial. És aquesta tecnologia tant trencadora com sembla? La podran aplicar les empreses del nostre país? Quina podria ser la inversió inicial necessària d'una empresa per poder començar a gaudir dels avantatges d'aquesta tecnologia?

Aquest treball intentarà respondre totes aquestes preguntes a la vegada que es desenvoluparà un programa d'ajuda a la presa de decisions empresarials utilitzant intel·ligència artificial.

El treball es divideix en una primera part explicativa que situa el lector amb l'estat de l'art de la tecnologia actual així com amb diverses solucions comercials ja existents.

La segona part del treball explica el desenvolupament d'un programa desenvolupat amb Python que permet predir si una oportunitat de negoci d'una empresa s'acabarà convertint en una venda o no. Per tal de realitzar aquestes prediccions s'utilitza una base de dades amb totes les oportunitats d'una empresa fictícia. Normalment aquesta eina s'anomena CRM (Customer Relationship Management).

Amb els jocs de dades que es disposen s'ha aconseguit un model que aconsegueix predir correctament en un 93% dels casos si la oportunitat concreta s'acabarà convertint en una venda o no.

Durant l'explicació d'aquest programa s'han intentat explicar els diferents conceptes utilitzat directa o indirectament (a través de diverses llibreries).

Finalment s'exposa el cost aproximat que podria tenir un projecte similar a la realitat, les conclusions del treball i proposes de desenvolupaments futurs.

Sumari

1. Glossari

AI - Intel·ligència artificial	Habilitat d'una màquina (ordinador) de realitzar tasques normalment associades amb essers intel·ligents
API	Interfície de programació d'aplicacions. És un conjunt de subrutines, funcions i procediments que poden ser utilitzades per un altre software com una capa d'abstracció
CPU	Unitat central de procés. Part central d'un ordinador encarregada de processar les dades
CRM	Sistema que gestiona les relacions entre una empresa i els seus clients (actuals i potencials)
GPU	Chip lògic programable especialitzat en imatges, animacions i vídeos. També molt utilitzat per realitzar processos que requereixen d'una gran potència de càlcul.
Machine Learing	Subcamp de la informàtica que permet que els ordinadors "aprenguin"
TPU	Unitat processadora de tensors. És un circuit integrat desenvolupat per Google específicament per accelerar el desenvolupament en intel·ligència artificial
Xarxes Neuronals	Sistema computacional modulat basant-se en el cervell i el sistema nerviós humà

2. Prefaci

2.1. Origen del projecte

Actualment la intel·ligència artificial (AI) surt de manera recurrent en publicacions de premsa i als diferents medis de comunicació. Això conjuntament amb les expectatives que genera desperten dubtes que aquest treball pretén poder respondre.

La gran majoria d'empreses utilitzen intel·ligència artificial? Què és exactament la intel·ligència artificial i per què es pot utilitzar? El futur per aquesta tecnologia és tan disruptor com es promet?

2.2. Motivació

La principal motivació per a la realització d'aquest treball va estar aprendre sobre el món de la intel·ligència artificial i concretament sobre les xarxes neuronals. A partir d'aquest interès i del fet que tot hi que es parlava molt del tema no es veien aplicacions directes van portar a l'elecció d'aquest tema.

Finalment al haver estat estudiant Enginyeria Industrial a la UPC i Administració i Direcció d'Empreses a la UOC, també es buscava un treball on es poguessin construir i aplicar eines (amb les coneixements adquirits a l'enginyeria) al món de la direcció empresarial. D'aquí que el treball es centri en com facilitar la presa de decisions empresarials utilitzant la intel·ligència artificial.

2.3. Requeriments previs

Per tal de comprendre els apartats introductoris d'aquest treball no és necessari disposar de cap coneixement previ. Tot hi això, una part important del treball és codi Python i explicacions matemàtiques. És per això que uns coneixements bàsics de programació així com de àlgebra i càlcul seran necessaris per treure el màxim profit d'aquesta lectura.

3. Introducció

3.1. Objectius del projecte

El primer pas per poder afrontar les qüestions prèviament esmentades en el prefaci serà entendre el panorama actual sobre la Intel·ligència artificial. S'exploraran les diverses solucions existents que apliquin aquesta tecnologia, fent especial èmfasis al que estan més centrats en la presa de decisions empresarials.

En segon lloc s'aprofundirà en la construcció d'una solució específica. Per tal de realitzar aquesta tasca aquesta part del treball es desenvoluparà sobre la plataforma de "Colab Notebooks" [1] de Google que ens permetrà desenvolupar directament des del navegador (permetent així desenvolupar des de diversos ordinadors incloent un Chromebook). S'utilitzarà el framework de "Tensorflow" [2] i Eager [3] que ens facilitaran molt les tasques de programació.

Finalment també s'utilitzarà les llibreria "Keras" que ens permetrà definir el model amb més facilitat.

Totes aquestes eines i llibreries s'exposaran amb més detalls en apartats posteriors del treball.

3.2. Abast del projecte

L'abast del projecte és realitzar una introducció genèrica en el món de la intel·ligència artificial i exposar diferents solucions comercials existents per tal de situar al lector.

Finalment el treball es centrarà en el desenvolupament d'un programa que llegint la base de dades d'un CRM (joc de dades de prova) aconsegueixi fer prediccions sobre si una oportunitat concreta acaba esdevenint una venta o no amb una precisió considerable.

Durant l'explicació d'aquest programa s'entrarà amb més detall amb els conceptes matemàtics i lògics que s'utilitzen ja sigui de forma directa o indirecta (utilitzats per alguna de les llibreries usades).

4. Intel·ligència artificial

4.1. Definició d'intel·ligència artificial

La primera menció de la que es té constància del terme "Intel·ligència Artificial" va ser escrita l'any 1956 per John McCarthy. En aquell cas es van utilitzar aquests mots en el títol "Dartmouth Summer Research Project on Artificial Intelligence". En aquesta trobada es van ajuntar diversos investigadors de camps diferents per tal de teoritzar sobre el concepte de "màquines pensants".

Des d'aquests inicis, el nombre de definicions ha anat augmentant conjuntament amb el nombre d'aplicacions que s'han trobat a aquesta tecnologia. Actualment les definicions més acceptades en la gran majoria de diccionaris defineixen la intel·ligència artificial com un sub-camp de la informàtica en el que es programen màquines per realitzar tasques que normalment requeririen de certa "intel·ligència humana".

Tot hi això aquesta definició pot variar depenent del propòsit que es tingui per la intel·ligència artificial com a tal. Alguns exemples podrien ser els següents:

- Construir sistemes que pensin de forma similar al humans.

- Construir sistemes que funcionin, sense que necessàriament entenguin repliquin el raonament humà.

- Utilitzar el raonament humà com a possible model sense que aquest sigui l'objectiu final.

D'aquestes tres definicions o objectius, el més utilitzat en l'actualitat és el tercer.

4.2. Identificació i selecció de la literatura

La intel·ligència artificial tal com l'entenem en l'actualitat és una temàtica en contínua evolució. És per aquesta raó que no existeix un manual o document que sigui acceptat per tothom. Tant és així que una gran part d'aquesta tecnologia encara està en fase de desenvolupament.

Per identificar fonts d'informació adequades s'ha mantingut la separació descrita a l'apartat anterior: Generalista i Tècnica.

La part generalista d'aquest treball pretén donar una visió al lector de l'estat de l'art d'aquesta tecnologia, de les aplicacions d'aquesta en l'actualitat i del potencial futur que pot tenir. Així com incidir en el possible impacte que pot tenir aquesta tecnologia per a les PIMES del nostre país. Per aquesta primera part s'han seleccionat fons reconegudes en el món dels negocis que utilitzen un llenguatge poc tècnic (fent referència a que no descriuen els models informàtics o matemàtics). Alguns exemples d'aquestes fons d'informació que s'han utilitzat poden ser "Forbes" , "Inc.com" o "Entrepreneur".

La part tècnica d'aquest treball pretén incitar el desenvolupant d'un programa similar al que les PIMES Catalanes podrien desenvolupar. Les opcions actuals per les diverses PIMES són, o bé utilitzar algunes de les solucions existents o desenvolupar-ne de personalitzades. Alguns exemples de fons d'informació que poden facilitar el desenvolupament de solucions personalitzades són el curs "Deep Learnig" [4] donat per "Andrew Ng" i la documentació oficial de "TensorFlow" [5] entre altres.

Aquest format de separació entre la part més genèrica i l'exemple més pràctic permetrà presentar degudament la tecnologia i a la vegada desenvolupar una solució concreta.

4.3. Història i tendències de la intel·ligència Artificial (AI)

Depenent de la literatura que es consulti l'inici de la intel·ligència artificial varia des de la primera màquina de calcular fins a el primer "script" capaç de realitzar una feina que realitzava anteriorment una persona humana.

A l'actualitat les diverses tendències dins de la intel·ligència artificial queden molt ben representades en la següent figura següent presentada per Gartner "The AI Cycle".

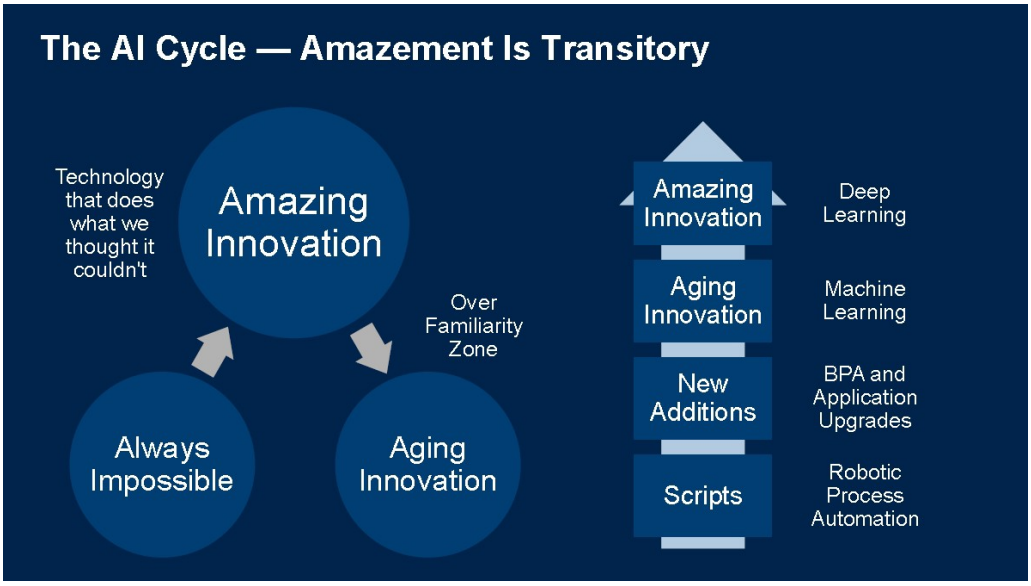


Figura 1 - Gartner - The AI Cycle

La part esquerra de la figura mostra els diversos estats pels que passa una nova tecnologia. Primer hi ha una tasca o objectiu que es considera impossible. A continuació la tecnologia aconsegueix complir aquesta tasca (per exemple que un ordinador guanyés a un gran mestre d'escacs en el seu moment). És en aquest punt en el que es troba la intel·ligència artificial actualment. Està aconseguint desenvolupar tasques que fins ara semblaven impossibles i és considerada una gran eina per el futur. A mesura que aquesta tecnologia vagi essent més coneguda el públic s'anirà familiaritzant amb aquests resultats i s'entrarà a la tercera rodona "Aging Innovation".

La gràcia d'aquesta figura també resideix en la part dreta d'aquesta on es descriuen 4 corrents diferents d'intel·ligència artificial que s'estan aplicant actualment. Les ordena des

de les més “senzilles” o les que requereixen i suposen una transformació menor fins a les que poden suposar una major transformació.

Així doncs, s’inicia amb petits “scripts” que permeten automatitzar processos molt definits i delimitats i es va evolucionant fins arribar a “Deep learning”, exemple del qual es desenvoluparà en aquest treball.

4.4. Eines comercials existents que utilitzen AI

Per a la gran majoria de PIMES el màxim valor que pot aportar d'intel·ligència artificial actualment és la millora de la productivitat.

Per aconseguir aquesta millora a través de d'intel·ligència artificial en la majoria de casos primer s'haurà de disposar de tota la informació rellevant de l'empresa de forma informatitzada.

Aquesta informatització de la informació es pot aconseguir implementant diversos sistemes, per exemple un ERP (Un sistema de planificació de recursos empresarials). Un dels majors exponents comercials seria el software que comercialitza l'empresa SAP però existeixen altres exemples (QAD, Microsoft Dynamics...).

Una vegada l'empresa es senti còmode utilitzant aquests sistemes es podrà començar a plantejar l'idea d'implementar un sistema amb intel·ligència artificial per tal que aquest pugui ajudar o automatitzar algunes de les decisions que s'han de prendre.

La primera eina que segurament es plantejaran utilitzar aquestes empreses serà el mateix ERP ja que les principals solucions comercials d'aquests softwares disposen de complements basats en intel·ligència artificial.

Una altra de les possibles eines que algunes PIMES podrien plantejar-se utilitzar per fer uns millors anàlisis d'aquesta informació i intentar extreure conclusions i relacions entre dades seria la solució "Watson" de I.B.M. [6]. Aquesta eina facilita l'aplicació d'intel·ligència artificial a diferents tasques.

I.B.M organitza les capacitats del seu software en tres apartats diferents:

- “Watson Discovery” permet explorar bases de dades i treure respostes i patrons.

- “Watson Assistant” ajuda a desenvolupar chatbots o agents virtuals en diversos canals per a facilitar per exemple la comunicació amb els clients.

- “Watson Visual Recognition” permet classificar i catalogar contingut visual. Aquesta funcionalitat es podria aplicar per exemple en els magatzems per detectar productes defectuosos.

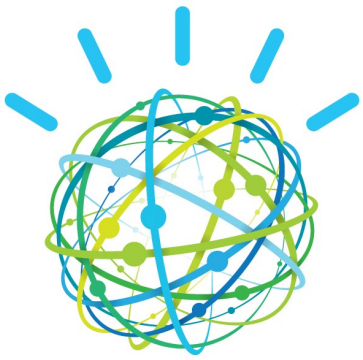


Figura 1- Watson I.B.M

D’una manera similar a Watson de I.B.M, TensorFlow també permet desenvolupar funcionalitats similars. Aquesta serà l’eina utilitzada per a la part més tècnica d’aquest treball.

A part d’aquestes eines també existeix software específic que pot ser utilitzat per empreses (ja siguin grans o petites) que utilitza intel·ligència artificial per tal d’aportar un major valor afegit.

A continuació es citaran alguns exemples:

- X.ai [7] : Un assistent personal que està entrenat a gestionar i coordinar les nostres reunions. Aconsegueix això amb un llenguatge natural (converses amb les altres parts) i es poden escollir dos perfils diferents l’Amy o l’Andrew.

- Crystal [8]: Es tracta d’una plataforma que permet veure el perfil de personalitat dels teus clients i per tant poder adaptar millor les nostres campanyes de comunicació amb ells.

- Tamr [9]: Permet integrar diverses fons d'informació internes de la companyia i posteriorment gràcies a la intel·ligència artificial detectar àrees de millora.

- Conversica [10]: Es tracta d'un assistent virtual que manté diverses conversacions amb possibles clients utilitzant el correu electrònic. Una vegada detecta una oportunitat real de venda pot avisar a un comercial perquè aquest l'aconsegueixi tancar.

- Recorded Future [11]: Aquest software protegeix a l'empresa en qüestió de diverses amenaces de ciberseguretat. Gràcies a utilitzar intel·ligència artificial és capaç d'analitzar un gran volum de dades de l'empresa de forma contínua per tal de prevenir qualsevol atac.

Tot hi mencionar aquests exemples específics centrats en ambients empresarials no s'han de descarar exemples més quotidians com ara l'assistent personal d'Apple "Siri" o algunes funcions de Google o Excel que auto-completen informació utilitzant aquesta tecnologia.

Està clar que la intel·ligència artificial és cada vegada més present a les nostres vides i als nostres llocs de treball però quin pot arribar a ser l'impacte d'aquesta tecnologia?

4.5. Disseny i construcció d'eines personalitzades

Dins del disseny i la construcció d'eines personalitzades s'han analitzat les diverses alternatives. Si es pretén crear un nou algoritme des de zero i controlar tots i cada un dels paràmetres es pot realitzar una programació amb Python o Matlab entre altres.

La bibliografia seguida per a la realització d'aquest treball així l'educació rebuda a la universitat ha més centrada amb Python. Així doncs, donada la flexibilitat i la fàcil comprensió d'aquest llenguatge, Python ha estat el llenguatge escollit.

Un desenvolupament des de zero d'un programari d'intel·ligència artificial amb Python requereix uns coneixements matemàtics previs considerables. Per tal de poder opinar d'aquest tema amb coneixement de causa es va seguir el primer capítol d'un curs online que exposa els diversos conceptes de Deep Learning i l'aplicació aquests amb Python [4].

Durant aquest període de temps es van aconseguir realitzar diversos programets petits però també es va poder observar la complexitat que podien arribar a assolir aquests projectes.

Per tal de facilitar la tasca de desenvolupament de software i donar algunes funcions ja realitzades existeixen eines com Tensorflow.

Tensorflow és una llibreria oberta que permet la realització de computacions complexes a una gran velocitat. Aquesta llibreria va estar desenvolupada per els enginyers de l'equip de Google Brain.

Una vegada seleccionat Python com a llenguatge i Tensorflow com a llibreria se'ns obren multitud de possibilitats. A la mateixa pàgina web ens exposa algunes de les possibilitats d'aquesta llibreria:

- Imatges
- Seqüències de text o àudio
- Anàlisi de informació i classificació
- Altres aplicacions matemàtiques no considerades com intel·ligència artificial

Tot i que una gran quantitat d'aquestes possibilitats poden ser aplicades en un entorn empresarial el grup que s'ha seleccionat per el possible major valor que pot aportar és el d'anàlisi d'informació i classificació d'aquesta.

5. Predicció del resultat de les entrades d'un CRM

5.1. Objectiu

Com que aquest treball tracta d'aplicar la intel·ligència artificial per a facilitar el procés de presa de decisions empresarials s'ha buscat un escenari on la informació estigui digitalitzada i preparada per analitzar amb un programa informàtic. També s'ha considerat quines possibilitats tindrien un impacte major en quan a retorn econòmic per part de l'empresa.

Una de les decisions més importants que han de prendre les empreses és on destinen els seus recursos per obtenir un major rendiment.

En el cas dels equips de ventes aquestes decisions la majoria de vegades es prenen sobre la base de dades d'un sistema CRM (Customer Relationship Management). Aquest sistema disposa d'una base de dades amb totes les oportunitats de venda (ja sigui d'un producte o servei) als diferents clients. No totes les oportunitats de venda acaben succeint, és per això que la decisió d'on es destinen recursos de personal per assegurar que aquestes oportunitats succeeixin és tant important.

En aquest cas s'ha escollit crear un programa que fent un anàlisi d'un extracte de les diverses oportunitats de negoci passades (ja siguin vendes o projectes) i del resultat que es va obtenir amb aquestes (venda, realització del projecte o pèrdua del client o projecte) pugui predir el resultat de vendes o projectes futurs.

Aquesta capacitat pot ser molt interessant per molts negocis que funcionen amb un sistema d'oportunitats "d'embut" on les oportunitats passen per diversos estats mentre es van consolidant. Si aconsegueixes predir quines d'aquestes oportunitats acabaran passant pots centrar els esforços majoritàriament en aquestes i acabar obtenint un major rendiment dels teus recursos (personal de vendes en aquest cas).

Per la realització d'aquest treball s'han utilitzat diversos sets de dades obtinguts de la pàgina web "Data World" [12] citada en la bibliografia.

Donats els bons resultats obtinguts no es descarta utilitzar un programa molt similar sobre CRM's d'empreses reals.

5.2. Estructura del projecte

5.2.1. Colab Notebook

Tal com s'ha esmentat anteriorment el model es dissenyarà dins d'un Colab Notebook.

Colab Notebooks també anomenat Colaboratory és un projecte de Google basat en "Jupyter notebook". Aquesta eina permet escriure codi, text i imatges en un mateix document.

Així doncs tal com es pot veure en la següent figura, el document es separa en apartats amb text i/o imatges i apartats amb codi executable.

Dins del codi es pot executar codi python o directament des de la línia de comandes (introduïnt un signe "!" a l'inici).

Els trossos de codi en un Colab Notebook es separen el cel·les i cada cel·la es pot executar independentment de la anterior. Cal destacar que las variables definides es poden accedir des d'una cel·la executada posteriorment.

Es va seleccionar aquest entorn de desenvolupament degut a la gran flexibilitat que ens ofereix ja que es pot programar i testejar el codi des de casi qualsevol dispositiu que disposi d'un navegador. Facilitant així que una part important d'aquest treball pugui ser desenvolupada amb un Chromebook sense quasi memòria ni capacitat de processament.

El fet que el codi s'executi online, sense consumir recursos de màquina (podent executar de manera gratuïta amb CPU o GPU) també ha estat un punt determinant.

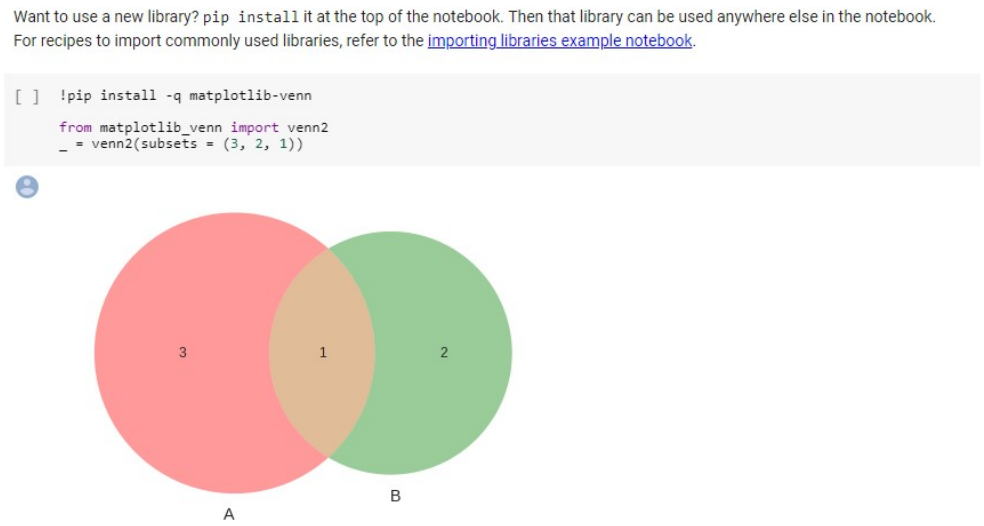


Figura 3- Watson I.B.M

5.2.2. TensorFlow

TensorFlow és una biblioteca de software lliure que permet realitzar càlculs numèrics amb una gran efectivitat. A més a més és suficientment flexible per ser desplegada en entorns executats per CPU (Central Processing Unit), GPU (Graphics Processing Unit) o TPU (Tensor Processing Unit). Essent aquest útil entorn bastant enfocat a córrer dins l'entorn que google proporciona online per les empreses (Google Cloud).

Aquesta llibreria va ser originalment desenvolupada per enginyers de la divisió d'intel·ligència artificial de google. Alguns dels seus majors avantatges són la rapidesa que té a realitzar càlculs i la flexibilitat que ofereixen les seves APIs a alt nivell així com la flexibilitat de realitzar modificacions a baix nivell per adaptar millor el teu model al problema concret.

A més a més TensorFlow és fàcilment instal·lable en qualsevol Colab Notebook.

5.2.3. Eager

L'execució de codi amb Eager forma part de la llibreria de TensorFlow.

Eager permet avaluar operacions immediatament sense la necessitat de generar graphs. Així doncs permet retornar directament valors sense construir el graph computacional i executar-lo després.

Eager està desenvolupat de manera que per defecte entén i processa les estructures de dades utilitzades en Python. També facilita la detecció d'errors ja que permet utilitzar les eines de "debugging" de Python.

5.2.4. Keras

Keras és una una llibreria i API que permet construir des d'un alt nivell xarxes neuronals. Està construïda amb Python i és capaç de executar-se dins d'un programa basat en TensorFlow.

L'objectiu de Keras és facilitar la experimentació, permetent així passar ràpidament del prototipatge a la producció. Tal com passa amb TensorFlow, Keras també ens permet executar el codi tant amb CPU com amb GPU.

5.3. Funcionament bàsic d'un model desenvolupat amb TensorFlow

Per tal de poder prosseguir amb l'explicació del model realitzat primer cal entendre alguns conceptes molt bàsics d'aquest.

El model que s'utilitzarà és un model que es separa en 3 fases importants: l'entrenament del model, la comprovació dels resultats i la resposta d'aquest.

Per tal de poder entrenar el model és necessita informació organitzada i la resposta que s'espera d'aquesta. Com més gran sigui aquesta base d'informació i més precisa més exacta serà la resposta de l'algoritme, en aquest cas una classificació.

L'exemple que ens posa la literatura de TensorFlow en aquest cas és una classificació en que a partir de dades d'un individu com ara l'educació, la situació sentimental, l'ocupació... Ens fa una predicció de si aquesta persona té un salari superior als 50mil dòlars americans.

Un programa com l'exposat anteriorment s'anomena un programa de classificació binaria (0,1). Ja que només té dos possibles respostes. O bé la predicció és que la persona té un salari superior als 50mil $ o bé aquest és igual o inferior.

Combinant diverses d'aquestes classificacions es pot aconseguir una classificació per segments (salarials en aquest cas).

5.4. Programa

En aquest apartat es descriuran les diferents parts del codi aportant també tots els conceptes teòrics necessaris.

5.4.1. Joc de dades

Per tal de poder predir si les oportunitats es convertien en ventes o no primer es necessitava un joc de dades d'un CRM lo suficientment extens com per poder aplicar deep learning. La primera intenció era utilitzar un joc de dades empresarial real però això va resultar més complex de lo esperat.

És per això que es va optar per buscar un joc de dades online. Finalment es va optar per el joc de dades "CRM data set" [12] de data world donat que la seva complexitat i estructura s'adequava als propòsits d'aquest treball.

Aquest joc de dades es divideix inicialment en 5 fitxers amb format .csv. Aquests fitxers són: accounts.csv, clicks.csv, products.csv, sales-teams.csv i sales-pipeline.csv. Essent aquest últim el més important amb més de 14000 oportunitats diferents.

Posteriorment es van analitzar els 5 fitxers i es va optar per descartar inicialment el fitxer clicks.csv donat que era més difícilment relacionable amb els altres. Posteriorment es van consolidar els altres 4 fitxers en un sol fitxer. Per cada oportunitat es va partir dels camps del fitxer sales-pipeline.csv i es van adherir els camps corresponents dels altres fitxers. Obtenint així, tota la informació de cada oportunitat en una sola línia (i tota la informació en una sola taula).

Finalment es van eliminar les oportunitats que encara resultaven obertes, quedant-nos només amb les que havien resultat en venta o havien estat descartades. Per facilitar més la programació també es van codificar les oportunitats que havien resultat en venta amb un 1 i les que s'havien perdut amb un 0. Una classificació similar es va realitzar amb la resta de variables (columnes).

Finalment es va ordenar de forma aleatòria i es va dividir en dos fitxers independents. El més extens d'aquest destinat a entrenar el model i el segon destinat a testejar-lo.

Seguidament els dos jocs de dades van ser pujats a la plataforma google drive donada la facilitat que aquesta ofereix per editar-los i de permetre que aquests siguin consultats per forns externes.

5.4.2. Instal·lar TensorFlow i importar les llibreries

Cal destacar que aquest model s'ha construït basant-se en models d'exemple trobats a la documentació oficial de TensorFlow i Keras.

El primer pas és instal·lar la última versió de TensorFlow.

En aquest cas el paràmetre "!" ens indica que aquesta línia de codi s'executarà des de la línia de comandes.

```
!pip install -q --upgrade tensorflow
```

Posteriorment s'importen els mòduls necessaris. En aquesta cel·la de codi cal destacar que algunes vegades dona error al executar-la si prèviament no s'ha reiniciat el "RunTime" del Colab Notebook.

```
from __future__ import absolute_import, division, print_function

import os
import matplotlib.pyplot as plt

import tensorflow as tf
import tensorflow.contrib.eager as tfe

tf.enable_eager_execution()

print("TensorFlow version: {}".format(tf.VERSION))
print("Eager execution: {}".format(tf.executing_eagerly()))
```

5.4.3. Importar i analitzar el joc de dades

En el moment de importar els diferents jocs de dades van aparèixer diversos problemes de format i de memòria cache ja que el programa seguia recordant l'últim fitxer pujat en lloc del nou.

Aquests dos problemes es van solucionar eliminant l'últim fitxer carregat a la memòria fitxer i imprimint per pantalla les primeres 5 línies del joc de dades per comprovar que efectivament el format era el correcte.

```
#Link del fitxer d'entrenament del model
train_dataset_url = "https://docs.google.com/spreadsheets/d/e/2PACX-
1vTKWa8RUXuZVQUkktlhkkB-
2OKFiqCp4Oq4vV_s7rnRLCq7TiYfOXAFZyWHvNiPTYfeK3NuDCoXZCnR/pub?output=csv"
 #train dataset

#Elimina l'últim fitxer carregat a la memòria cache
!rm /content/.keras/datasets/pub?output=csv

train_dataset_fp =
tf.keras.utils.get_file(fname=os.path.basename(train_dataset_url),
                                            origin=train_dataset_url)

print("Local copy of the dataset file: {}".format(train_dataset_fp))

#Només per informació
#Llegeix les primeres files del fitxer train_dataset_fp i els imprimerix
per pantalla per comprovar que té sentit
!head -n5 {train_dataset_fp}
```

A continuació es defineix la funció parse_csv, que com el seu nom indica modifica el format de cada línia que li arriba per passar-lo d'un format .csv a un format de llista amb el que Python pot treballar més fàcilment.

A més a més aquesta funció defineix valors per defecte per evitar tenir valors blancs que puguin fer que el nostre model deixi de funcionar. Entre aquests valors per defecte cal destacar "43305.". Aquest valor representa el 24 de Juliol de 2018 en el format utilitzat per Microsoft Excel. Aquesta data es va introduir manualment com la data en que es va penjar l'última versió del model de dades. Per tal d'introduir dates i que aquestes fossin fàcilment processades per TensorFlow i Keras com a nombres decimals es va utilitzar aquesta nomenclatura. És una data arbitrària que simplement serveix com a referència.

```
#Modifica el format del joc de dades d'un .csv a una llista interpretalbe
per python utilitzant la llibreria de TensorFlow (tf)
```

```
#Es defineixen els valors per defecte dels camps en cas que aquests
siguin buits
def parse_csv(line):
  example_defaults = [[0.], [0.], [0.], [0.], [0.], [0.], [43305.], [0.],
[0.], [0.], [0.], [0.], [0.], [0]]  # sets field types
  parsed_line = tf.decode_csv(line, example_defaults)

  # Els primers 13 camps (característiques) es convinen en un únic tensor
  features = tf.reshape(parsed_line[:-1], shape=(13,))

  # l'última columna és el resultat esperat.
  label = tf.reshape(parsed_line[-1], shape=())

 #La funció parse_csv retorna les característiques i el resultat (label)
com a llistes en format python
  return features, label
```

A continuació es defineix la variable “train_dataset” com el joc de dades d’entrenament. Aquest joc de dades s’adapta per poder ser posteriorment processat de la forma més eficient possible per el model.

Es descarta la primera fila ja que conté els títols de les diverses columnes, es posa cada fila en format de tensor, s’ordenen les diferents files aleatòriament i es defineix quantes files a la vegada es podran carregar en el programa. Aquest valor pot ser modificat i variar el percentatge d’encert del model.

```
train_dataset = tf.data.TextLineDataset(train_dataset_fp) #agafa el joc
de dades previament carregat i converteix en el format tractat per
tensorflow
train_dataset = train_dataset.skip(1)             # evita la primera fila
del joc de dades que conté els noms de les columnes
train_dataset = train_dataset.map(parse_csv)      # fa el parse de cada
una de les files
train_dataset =
train_dataset.shuffle(buffer_size=len(train_dataset_fp)+1000)  # posa les
files en ordre aleatori, el buffer ha de ser superior al joc de dades
train_dataset = train_dataset.batch(25) #Defineix el nombre de files que
es podran carregar a la vegada

# S'imprimeix a la pantalla el primer exemple de les característiques i
del resultat per evaluar si el format és el correcte
features, label = iter(train_dataset).next()
print("example features:", features[0])
print("example label:", label[0])
```

5.4.4. Definició del model

En aquest apartat s'utilitza la llibreria Keras que facilita la definició de models en xarxes neuronals.

En el següent sub-apartat (Teoria de models en xarxes neuronals) s'exposa amb més detall la teoria que Keras executa per nosaltres.

Es defineix un model amb 3 capes ocultes (hidden layers) que tenen respectivament 12,10 i 10 nodes.

Tal com és habitual en aquests models la funció d'activació és la ReLu (explicada en el proper apartat).

Així doncs el model proposat i simplificat tindria un aspecte similar al de la figura següent:

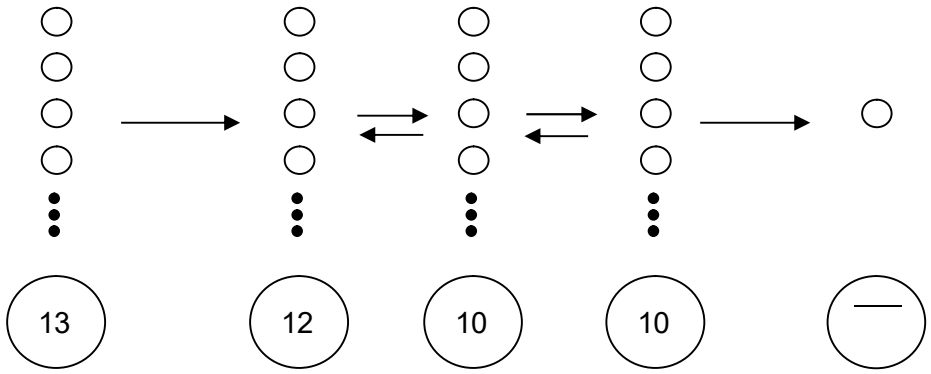


Figura 4- Model Simplificat

Com més neurones s'afegeixen més complex és el model i un major volum de dades és necessari per poder modelar-lo correctament.

```
#Es defineix un model amb 3 capes ocultes de 12, 10 i 10 nodes
respectivament. Amb una entrada de 13 característiques i una sortida de 2
possibilitats.
#Utilitzarem la funció ReLU  (Tal com s'ha après en els tutorials de
Andrew NG ) per les capes ocultes
model = tf.keras.Sequential([
  tf.keras.layers.Dense(12, activation="relu", input_shape=(13,)),  #
format d'entrada (13) és necessari
  tf.keras.layers.Dense(10, activation="relu"),
  tf.keras.layers.Dense(10, activation="relu"),
  tf.keras.layers.Dense(3)
])
```

El següent pas és definir les funcions de pèrdua i d'optimització.

Per la funció d'optimització s'utilitzarà GradientDescent ja incorporada i definida a TensorFlow. (Explicada amb més detall a l'apartat següent).

Un dels paràmetres més importants de tot el model és la constant d'aprenentatge. Aquesta constat és definida en aquesta apartat i és la que ens permet arribar a un òptim local millor o pitjor a una o altre velocitat.

Per aquest model la constant seleccionada (després de multitud de proves) ha estat: 0.000001)

```
#Es defineixen les funcions de pèrdua (Loss) i Gradient
def loss(model, x, y):
  y_ = model(x)
  return tf.losses.sparse_softmax_cross_entropy(labels=y, logits=y_)


def grad(model, inputs, targets):
  with tf.GradientTape() as tape:
    loss_value = loss(model, inputs, targets)
  return tape.gradient(loss_value, model.variables)

#Es defineix l'optimizer (GradientDecscentOptimizer) i s'especifica la
constant d'aprenentatge
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.000001)
```

5.4.4.1. Teoria de models en xarxes neuronals

Aquest apartat pretén representar de forma resumida la teoria existent sobre xarxes neuronals que s'ha utilitzat a través de la llibreria Keras. La literatura sobre el tema que es pot trobar online és extensa i no pot ser resumida en unes poques pàgines, això és només un petit resum. Si es vol indagar més en el tema es recomana el seguiment del curs de Andrew Ng. [4] citat a la bibliografia ja que és una de les fons que més es va consultar per a l'escriptura d'aquest apartat.

Les característiques de cada oportunitat (13 en total) estaran definides files en la matriu X. Així doncs, cada columna de la matriu X representarà una oportunitat amb totes les seves característiques. Per altre banda, el que s'intenta predir a partir d'aquesta matriu X són els resultats Y. Essent Y una matriu fila on cada columna representa el resultat obtingut en una oportunitat (0 o 1).

A partir d'aquest moment la teoria seguirà fent referència a una sola oportunitat (columna). Així doncs el vector de les característiques d'aquesta oportunitat quedarà definida amb la lletra "x" minúscula i el resultat esperat amb "y". S'ha de tenir en compte que l'explicació que seguirà a continuació hauria de ser posada en forma de matrius per realment poder ser aplicada de forma eficient en un programa.

A més a més per simplificar l'explicació aquesta només exposa el model amb una capa oculta. Aquest model pot ser extrapolat a més capes ocultes augmentat així la complexitat del model i la necessitat de disposar de més dades.

"Logistic regression"

El problema que es planteja en aquest treball és un problema que es pot atacar amb "Logistic regression". Aquest algoritme s'utilitza en problemes on l'aprenentatge és supervisat. És a dir, quan els resultats (Y) son o 0 o 1 i coneguts per el joc de dades. L'objectiu d'aquest algoritme és minimitzar l'error entre les prediccions i els resultats donats en el joc de dades d'entrenament.

Per de poder aplicar aquest algoritme es necessiten:

x : Vector de característiques d'entrada
y: resposta esperada del model (etiqueta)
w: vector de pesos (amb el mateix nombre d'elements que x)
b: variable límit

En aquest cas, per fer més fàcil la comprensió de l'explicació s'utilitzarà la funció "sigmoid" com a funció d'activació, tot hi que en el model s'ha utilitzat la funció ReLu.
La predicció que es vol obtenir es defineix com la variable ŷ. Essent ŷ:

$$\hat{y} = \sigma (w^T \cdot x + b)$$

Fórmula 1

on la funció sigmoid es defineix com:

$$s = \sigma (w^T \cdot x + b) = \sigma (z) = 1 /(1- e^{-z})$$

Fórmula 2

Tal com es pot observar a la fórmula anterior ($w^T \cdot x + b$) és una funció lineal com (ax + b). Com que el nostre objectiu és limitar la funció entre 0 i 1 (les dos úniques possible solucions) s'utilitza la funció "sigmoid".

A mode d'observació, si "z" té un valor gran i positiu, σ(z) tindrà un valor molt pròxim a 1 mentre que si aquest número és negatiu el valor de σ(z) serà pròxim a 0. Si z=0, aleshores σ(z)=0.5.

Diferents funcions d'activació:

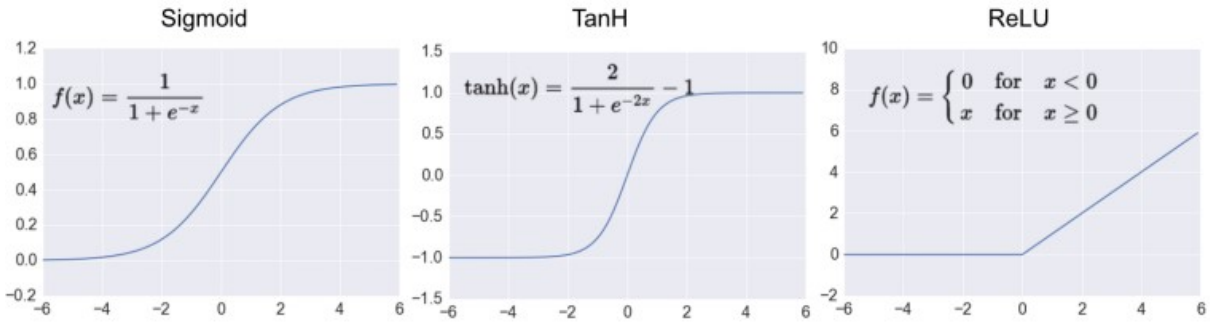


Figura 5 – Funcions d'activació

Funció de cost:

Recapitulant, el nostre objectiu és calcular ŷ. Tal com s'ha vist anteriorment per tal d'aconseguir-ho necessitarem definir els paràmetres "w" i "b". Aquest procés de definir els paràmetres s'anomena entrenament. A l'anotació anterior afegirem "(i)" corresponent a determinar que s'està parlant de l'oportunitat "i".

$$s = \sigma (w^T \cdot x^{(i)} + b) = \sigma (z) = 1 /(1- e^{-z(i)})$$

Fórmula 3

Donades m parelles de vectors x, y es vol aconseguir que ŷ sigui sempre el màxim semblant a y.

Funció error (Loss)

Per aconseguir aquest propòsit es defineix la funció error (Loss).
Aquesta funció mesura la discrepància entre la predicció $\hat{y}^{(i)}$ i el valor esperat $y^{(i)}$. Així doncs, computa l'error de la funció per un exemple específic (oportunitat en el nostre cas).

Una possible definició de la funció error (Loss) és:

$$L\left(\hat{y}^{(i)}, y^{(i)}\right) = -\left(y^{(i)} \cdot \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \cdot \log\left(1 - \hat{y}^{(i)}\right)\right)$$

Fórmula 4

Aquesta fórmula aparentment estranya permet fer que:

- Si $y^{(i)} = 1$: $L\left(\hat{y}^{(i)}, y^{(i)}\right) = -\log\left(\hat{y}^{(i)}\right)$ per lo que $\log\left(\hat{y}^{(i)}\right)$ i $\hat{y}^{(i)}$ hauran de ser valors pròxims a 0.
- Si $y^{(i)} = 0$: $L\left(\hat{y}^{(i)}, y^{(i)}\right) = -\log\left(1 - \hat{y}^{(i)}\right)$ on $\left(1 - \hat{y}^{(i)}\right)$ i $\hat{y}^{(i)}$ haurien de de ser pròxims a 0.

Funció cost (J)

La funció cost (J) es defineix com la mitjana de totes les funcions error (Loss) per tot el joc de dades d'entrenament. Es buscaran els paràmetres "w" i "b" que minimitzin la funció cost (J).

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}(i), y(i))$$

$$= -\frac{1}{m}\sum_{i=1}^{m}[\,y(i)\cdot \log(\hat{y}(i)) + (1 - y(i))\cdot \log(1 - \hat{y}(i))]$$

Fórmula 5

Gradient Descent

Com que el que es pretén buscar el valor mínim de la funció cost (J) es fa la derivada parcial d'aquesta amb les seves dos variables. El valor d'aquesta derivada parcial es pot representar com un pendent multidimensional que apunta cap a un mínim local (no té perquè ser el mínim global) de la funció cost.

Així doncs es defineix el nou paràmetre "w" com ell mateix menys una constant d'aprenentatge (α) multiplicada per la derivada parcial de "J" sobre "w". El mateix succeeix per "b".

$$w := w - \alpha\frac{\delta J(w,b)}{\delta w}$$

$$b := b - \alpha\frac{\delta J(w,b)}{\delta b}$$

Fórmula 6

Aquest procés es repeteix una gran quantitat de vegades. Al final si el paràmetre α ha estat seleccionat correctament i el nombre de nodes seleccionat ha estat correcte s'haurà

aconseguit reduir la funció cost fins a un mínim local, aconseguint així encertar un major nombre de vegades la predicció de ŷ.

S'ha de tenir en compte que aquest càlculs a la pràctica es realitzen de forma matricial per millorar en eficiència de càlcul. A més a més models més complexos (com l'estudiat en aquest treball) disposen de més capes ocultes de nodes que permeten augmentar encara més la fiabilitat de les prediccions del programa.

A la imatge següent es pot veure una gràfica un exemple en 2D. El nostre exemple seria multidimensional.

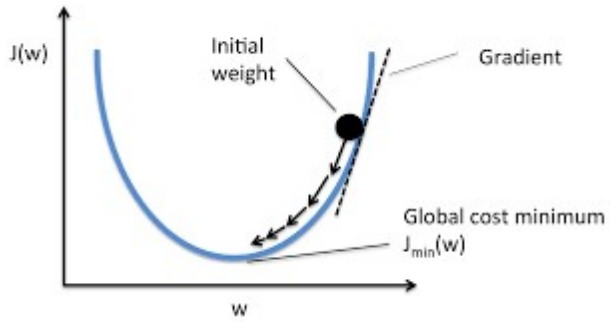


Figura 6 – "Gradient Descent"

Tal com s'ha comentat, aquest apartat pretén informar suficientment al lector per entendre com es computa el model i l'entrenament d'aquest. Aquest exemple també ens permet exemplificar les facilitats que ens aporta utilitzar una llibreria com Keras.

5.5. Entrenament del model

Per tal d'entrenar el model definit prèviament es segueixen els passos següents:

1. Iterar cada "epoch". Un "epoch" és una passada per tot el joc de dades.

2. Dintre de cada "eopoch" iterar per cada exemple en el joc de dades agafant les seves característiques i les seves etiquetes (resultats)

3. Utilitzant les característiques del joc de dades d'entrenament realitzar una predicció i comparar-la amb l'etiqueta (resultat). Mesurar la diferencia de la predicció i utilitzar-la per calcular la funció error (loss) del model que a la vegada serà necessària per calcular els gradients.

4. Utilitzar una funció (optimizer) per actualitzar els valors de les variables. En aquest cas es s'utilitzarà "Gradient Descent" que és el mateix que s'ha exposat a l'apartat de teoria anterior.

5. Visualitzar la funció Loss i el % d'encert per poder determinar si el model s'ha ja ha arribat al màxim nivell d'encert o si s'està sobre-entrenant.

6. Repetir aquest procés per cada "epoch"

Una variable important definida en aquest apartat es "num_epochs" que com el seu nom indica defineix el nombre de "epochs" que es realitzaran sobre el model de dades.

Un nombre massa petit de "epochs" donarà massa poca informació al model i aquest no estarà suficientment entrenat. Per altre banda, un nombre massa alt de "epochs" produirà un model sobre-entrenat amb el joc de dades d'entrenament. Això significa que serà molt bo predit els valors exactes del joc de dades d'entrenament però que segurament arran d'això serà més dolent a l'hora de predir altres valors.

En aquest cas (i després de realitzar multitud de proves) s'ha estimat que el nombre d'"epoch" que dona el millor resultat era 400.

```
# manté els resultats per poder imprimir-los
train_loss_results = []
train_accuracy_results = []
```

```
num_epochs = 400  #Es defineix el nombre de "epoch" que es volen executar

for epoch in range(num_epochs):
  epoch_loss_avg = tfe.metrics.Mean()
  epoch_accuracy = tfe.metrics.Accuracy()

  # Loop d'entrenament
  for x, y in train_dataset:
    # Optimitza el model
    grads = grad(model, x, y)
    optimizer.apply_gradients(zip(grads, model.variables),

global_step=tf.train.get_or_create_global_step())

    # Guarda el progrés
    epoch_loss_avg(loss(model, x, y))  # Afegeix el loss del batch actual
    # Compara l'etiqueta predia per el model amb la real
    epoch_accuracy(tf.argmax(model(x), axis=1, output_type=tf.int32), y)

  # Finalitza  l'epoch
  train_loss_results.append(epoch_loss_avg.result())
  train_accuracy_results.append(epoch_accuracy.result())

  #Imprimeix per pantalla el valor de la funció Loss i de la fiabilitat
del model cada 50 epoch.
  if epoch % 50 == 0:
    print("Epoch {:03d}: Loss: {:.3f}, Accuracy: {:.3%}".format(epoch,
```

Al final d'aquest entrenament el model aconsegueix endevinar correctament el si la oportunitat s'acabarà convertint en venta o no aproximadament un **96%** de les vegades sobre el joc de dades d'entrenament.

Els resultats obtinguts cada 50 "epoch" són els següents:

```
Epoch 000: Loss: 286.160, Accuracy: 76.739%
Epoch 050: Loss: 0.928, Accuracy: 91.576%
Epoch 100: Loss: 0.431, Accuracy: 93.963%
Epoch 150: Loss: 0.300, Accuracy: 94.952%
Epoch 200: Loss: 0.211, Accuracy: 95.600%
Epoch 250: Loss: 0.191, Accuracy: 95.839%
Epoch 300: Loss: 0.169, Accuracy: 96.146%
Epoch 350: Loss: 0.153, Accuracy: 96.180%
```

A continuació per facilitar la interpretació dels resultats s'han graficat. Aquesta gràfica és molt útil per veure per exemple si a partir d'algun moment es poden predir el nombre d'Epoch mantenint els mateixos resultats sobre el joc de dades d'entrenament o si es podrien fins hi tot augmentar i millorar els resultats.

```
#Ajuda a visualitzar el procés d'aprenentatge del model
fig, axes = plt.subplots(2, sharex=True, figsize=(12, 8))
fig.suptitle('Training Metrics')

axes[0].set_ylabel("Loss", fontsize=14)
axes[0].plot(train_loss_results)

axes[1].set_ylabel("Accuracy", fontsize=14)
axes[1].set_xlabel("Epoch", fontsize=14)
axes[1].plot(train_accuracy_results)

plt.show()
```

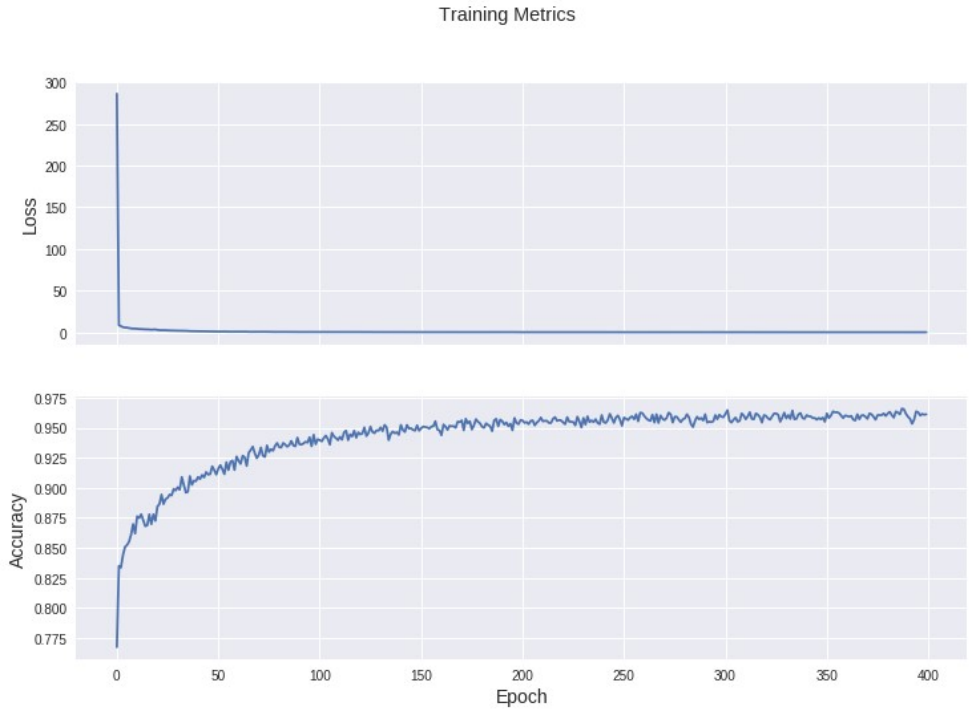


Figura 7 – Gràfiques entrenament

Tal com es pot observar la millora a l'inici del model és molt pronunciada, tot hi això a partir de l'"Epoch" número 100 aquesta millora queda bastant estancada.

5.6. Avaluar el model

Fins ara el model ha mesurat la seva efectivitat sobre el mateix joc de dades que ha utilitzat per entrenar-se. Això sol comportar que tingui un rendiment superior per aquest joc específic de dades que en la majoria de les dades.

És per aquest motiu que s'avalua el model amb un altre joc de dades (el joc d'avaluació o test). A l'inici del treball es va separar el joc de dades en dos, un d'entrenament i un d'avaluació de forma aleatòria.

Es segueix el mateix procés de carregar i donar el format adequat a les dades del joc de test i finalment es fa passar el joc de dades per el model i s'obté una fiabilitat del **93,599%.**

```
#Elimina el joc de dades emmagatzemat previament
!rm /content/.keras/datasets/pub?output=csv


#Joc de dades de prova (test)
test_url = "https://docs.google.com/spreadsheets/d/e/2PACX-
1vSiGMl9BH0JREhiW1BUqPgdaUjB88qfjWxpUYpoPxkr9DwX0XSpcQqlGz16g3gbod_-
lYV26SQCzGJX/pub?output=csv"

test_fp = tf.keras.utils.get_file(fname=os.path.basename(test_url),
                                  origin=test_url)

#Mateix procés realtizat anteriorment amb el joc de dades d'entrenament
test_dataset = tf.data.TextLineDataset(test_fp)
test_dataset = test_dataset.skip(1)
test_dataset = test_dataset.map(parse_csv)
test_dataset = test_dataset.shuffle(len(test_fp)+1000)
test_dataset = test_dataset.batch(32)


#Evaluate model on Test dataset
test_accuracy = tfe.metrics.Accuracy()

for (x, y) in test_dataset:
  prediction = tf.argmax(model(x), axis=1, output_type=tf.int32)
  test_accuracy(prediction, y)

print("Test set accuracy: {:.3%}".format(test_accuracy.result()))
```

```
Test set accuracy: 93.599%
```

5.7. Fer prediccions

Per fer prediccions es podrien entrar jocs de dades complets com en el model de dades o directament muntar una interfície d'usuari que permeti entrar dades d'una en una amb més facilitat.

Aquesta component no s'ha desenvolupat en aquest treball ja que no s'ha considerat necessari per el fet que fins ara el model s'ha desenvolupat sobre un joc de dades de prova i per tant no té utilitat pràctica directa.

5.8. Organització temporal

Una de les principals motivacions per realitzar aquest treball va ser l'interès per aprendre més sobre el camp de la intel·ligència artificial. Concretament sobre el camp de les xarxes neuronals.

Per tal d'aprendre més en detall vaig realitzar els dos primers apartats del curs de "Coursera" sobre xarxes neuronals impartit per Andrew Ng. [4].

Aquests dos primers apartats es centraven en anàlisis d'informació organitzada i no lineal. És a dir, informació estructurada.

Completar aquests dos apartats i entendre el funcionament bàsic d'una xarxa neuronal així com estructurar-les ha estat bàsic per posteriorment poder-ne explicar el funcionament d'una forma simplificada en aquest treball.

La primera intenció era desenvolupar tot el programa íntegrament amb Python carregant només llibreries com NumPy (facilita els càlculs de vectors). Tot hi això quan es portaven dos setmanes de desenvolupament es va començar a investigar la possibilitat de desenvolupar aquest programa a Colab Notebooks. Aquesta possibilitat permetia desenvolupar des de diversos dispositius amb facilitat, inclús si aquests tenien un sistema operatiu més limitat com pot ser Chrome OS.

Arran d'aquest canvi de direcció es va explorar la possibilitat d'utilitzar TensorFlow, cosa que va acabar repercutint en utilitzar Eager i Keras per facilitar la feina de desenvolupament.

Així doncs, els primers mesos van ser purament per començar a entendre el tema i seguir els cursos online. Posteriorment es va moure el desenvolupament a Colab Notebooks i finalment s'ha realitzat el redactat del treball.

5.9. Dificultats i obstacles superats

Una de les dificultats va ser aconseguir un joc de dades suficientment complet i complex sobre el que desenvolupar aquest treball.

La primera intenció era utilitzar el joc de dades del CRM que actualment utilitza una multinacional del sector tecnològic. Donada la dificultat d'obtenir aquest joc de dades es va optar per buscar-lo online. Després de diversos intents es va acabar trobant el joc de dades actual que ha permès desenvolupar aquest treball.

L'adaptació del joc de dades per poder ser utilitzat per el programa també va comportar més temps del que es tenia planificat ja que van sorgir diversos problemes amb els valors per defecte, les dates i el tipus de valors. Finalment tots els problemes van ser corregits i adreçats d'una manera o altre.

El següent seguit de dificultats va ser de caire tecnològic. Durant bastant de temps es va tenir un error en el codi en el que aparentment el format de les diverses columnes era incorrecte. Tot hi adaptar el joc de dades, l'error es seguia repetint. Al final es va detectar que l'error provenia del "cache". L'antic joc de dades es quedava emmagatzemat i no es modificava.

Un altre problema va sorgir quan sense una raó aparent el codi va deixar de funcionar d'un dia per l'altre. Finalment es va descobrir que el mode d'execució del codi s'havia canviat de "GPU" a "CPU". Tot hi que teòricament tant TensorFlow com Keras suporten ser executats amb "GPU", el codi no funcionava correctament.

El programa acaba arribant en òptims locals en un espai multidimensional. Així doncs, el punt d'inici des d'on es comença a aplicar l'estratègia de seguir el major gradient acaba determinant a quin mínim local s'arriba. Això ha suposat una dificultat per el fet que com que el punt d'inici és aleatori en codi proposat, és difícil replicar un bon resultat. Per tenir més detalls sobre aquest punt es recomana llegir l'apartat d'aquest treball sobre possibles futures millores.

5.10. Possibles futures millores

Les dos millores més importants que es podrien aplicar en aquest programa serien:

- Llegir dades amb diferents formats: Actualment el joc de dades s'ha manipulat i s'han codificat totes les dades a nombres decimals per facilitar-ne el tractament. Si es modifiqués el codi per tal que acceptés diferents tipus de variables això augmentaria la flexibilitat dels usuaris.

- Un segon pas que milloraria encara més la flexibilitat dels usuaris seria fer que no s'hagués de posar de forma manual el nombre de característiques (paràmetres) que s'estan carregant en el programa.

Combinant aquestes dos millores l'usuari seria capaç de pujar un fitxer amb format .csv amb un nombre indefinit de columnes amb característiques i on la última columna contingués el resultat en format binari (0 i 1 com a nombres enters).

La segona iteració de millores que es proposen serien en quant a rendiment. Tal com s'ha comentat anteriorment, el fet que el model obtingut (i l'eficàcia d'aquest) depengui del joc d'oportunitats d'inici dificulta la feina d'avaluació sobre si la programació del model ha estat la correcte o no.

Realitzar un bucle sobre tot el procés (incloent l'avaluació amb el joc de dades de test) permetria buscar quin és el punt d'inici amb una major precisió en quant a predicció. Aquest punt es podria emmagatzemar i acabar calculant el model sobre aquest.

A part del punt d'inici, altres paràmetres com poden ser el nombre de capes, el nombre de nodes en cada capa, el nombre de "epoch", la constant d'aprenentatge o el nombre d'oportunitats que es carreguen cada vegada son algunes de les variables que s'han determinat de forma arbitraria. Ja sigui per prova i error o per guies proporcionades per experts online. Si es disposa d'un codi on el pas anterior estigui ja realitat (existeixi un bucle per determinar millor òptim local entre un seguit de possibilitats) es podria realitzar analitzar aquesta millora com un problema d'optimització. Si ja tens els jocs de dades d'inici, tens un seguit de variables (nº nodes, nº capes...), una "funció" execució del model, que et dona un resultat (encert del model) que vols intentar maximitzar.

Aquesta segona iteració de millores segurament requeririen d'un temps de computació considerable, per això és tant favorable el fet d'haver fet el desenvolupament inicial

d'aquest programa per tal que pugui ser executat directament (i gratuïtament) per els servidors de Google.

Finalment el tercer pas seria fer encara més accessible aquest programa per un usuari que no estigui familiaritzat amb el codi. És a dir, crear una interfície gràfica on l'usuari pogués pujar el seu fitxer, entrar alguns paràmetres i demanar que es generi el model. Una vegada aquest usuari tingués un model generat, aquest podria enviar dades al model i aquest li retornaria la seva predicció.

Per a la realització d'aquest tercer pas són necessaris els dos primers.

Si es poguessin realitzar les tres millores suggerides el programa es podria comercialitzar per tal que diversos usuaris poguessin introduir els seus jocs de dades on el resultat sigui un nombre definit de nombres enters (0 i 1 a l'inici). Posteriorment aquest usuari podria obtenir prediccions del model generat amb les seves pròpies dades. Si es disposes d'un nombre d'usuaris suficientment elevat es podrien utilitzar les prediccions en quant a nombre de capes, nodes i constants d'aprenentatge per accelerar la generació del model òptim.

Aquestes millores no s'han desenvolupat en aquest treball degut a que s'ha preferit entendre el funcionament intern del model i aconseguir un model amb una fiabilitat acceptable (93,6%).

6. Pressupost

El pressupost d'aquest projecte consisteix bàsicament d'hores de desenvolupament ja que els costos operatius del programa són casi nuls i no s'ha requerit de cap llicència de pagament.

A part de les hores dedicades en el treball s'han comptabilitzat algunes altres despeses agrupades i finalment l'amortització de l'ordinador i equipament ofimàtic suposant que aquest tingués una vida útil de només un any.

En aquest pressupost no s'han incorporat hores d'aprenentatge autònom que han estat indispensables per tal de desenvolupar aquest treball.

Tasques	Professional	Duració (h)	Cost (€/h)	Cost (€)
Analitzar el problema a tractar	Analista de negoci	8	25	200
Escollir la millor plataforma i eines per atacar el problema	Arquitecte	24	45	1080
Estructurar plan de desenvolupament	Cap de projecte	4	50	200
Programació	Desenvolupador	150	25	3750
Testejar el programa i els resultats	Desenvolupador	8	25	200
Aplicar millores i corregir errors	Desenvolupador	75	25	1875
Redacció de l'informe (treball)	Cap de projecte	50	50	2500
Preparar presentació i presentar el projecte finalitzat	Cap de projecte	8	50	400
Total		**327**		**10205 €**

Altres costos	Cost total	Vida útil (h)	Temps d'us (h)	Cost (€)
Paper, impressió, electricitat…	18			18,00
Amortització de l'ordinador i altres components	1000	5*35=1,825	327	179,18
Total				**10402,18 €**

Tal com es pot veure a la taula anterior el cost total d'aquest projecte si fos realitzat per professionals seria d'aproximadament 10.402€.

El cost per hora de cada professional ha estat obtingut a través de plataformes online com "Glassdoor" i de consultar a professionals amb posicions similars. Aquests preus podrien variar considerablement depenent de l'experiència del professional i la ubicació geogràfica d'aquest.

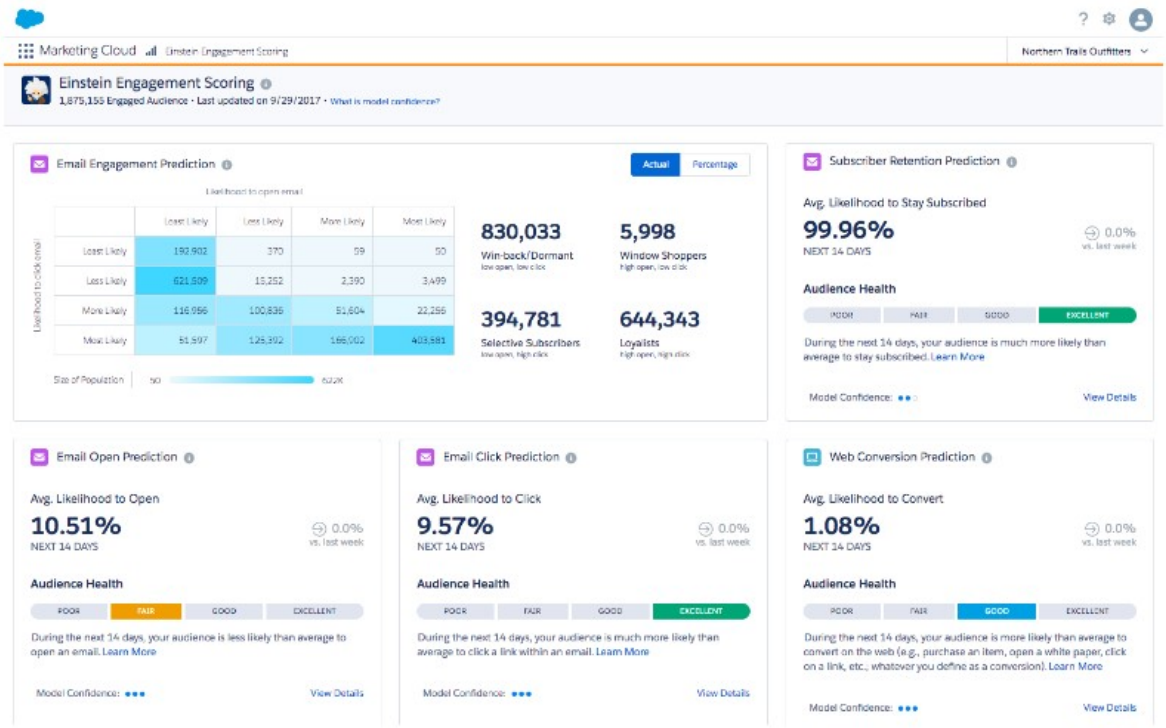
El nombre d'hores mostrat a la taula és el nombre d'hores aproximat que s'ha tardat per realitzar les diferents tasques d'aquest treball. Segurament un professional més experimentat les podria realitzar amb molt menys temps.

7. AI als CRM comercials actuals

Actualment, els 4 proveïdors de CRMs amb més quota de mercat són: Salesforce, Oracle, SAP i Microsoft [13].

Aquestes 4 solucions representen aproximadament el 50% del mercat. La majoria d'aquestes comencen ja a aplicar solucions que aporten un valor extra als seus clients basades en Intel·ligència artificial.

Per exemple, tal com es pot veure a la imatge següent, SalesForce proporciona als seus clients prediccions sobre la possibilitat de retenir un client, la probabilitat de que aquest obri un correu, que faci click...



Aquest és només un exemple de les múltiples aplicacions que existeixen. Les grans empreses han estat adquirint tecnologia i coneixement en el sector d'intel·ligència artificial comprant StartUps especialitzades o cultivant-lo internament.

Tot hi això les seves solucions són tancades i exclusives per els seus productes. La solució que proposa aquest treball és més àmplia ja que amb petites modificacions permetria donar prediccions sobre jocs de dades completament diferents.

8. Impacte ambiental

Al tractar-se d'un projecte de desenvolupament de software l'impacte ambiental és mínim. Aquest és bàsicament la electricitat consumida i la potencial generació de CO_2 necessària per tal de produir aquesta.

A la taula següent es poden trobar les emissions de CO_2 generades al realitzar aquest treball:

Elements	Temps	Potència consumida [W]	Energia consumida [kWh]	CO_2 consumit [Kg]
Ordinador	327	120	39,24	25,51
Il·luminació	250	89	22,25	14,46
		Total	**61,49**	**39,97**

L'ordinador utilitzat majoritàriament per realitzar aquest treball ha estat un Asus N56. La il·luminació ha estat proporcionada per una làmpada de 89W.

La Unió Europea considera que per tal de produir 1kWh es necessiten 0,65 kg de CO_2.

No s'han considerat el consum dels servidors ja que l'ús ha estat casi negligible. Si es portés un sistema similar en un entorn de producció, aquest ús s'hauria de considerar.

9. Conclusions

La intel·ligència artificial és un camp en expansió on ja existeixen diverses solucions comercials. Tot hi això encara no s'ha arribat a aconseguir els resultats esperats. En aquest treball s'han exposat diverses solucions comercials així com alternatives per desenvolupar programari més personalitzat.

S'ha investigat el camp de les xarxes neuronals en dos besants, la més teòrica i la més pràctica per entendre com funcionen les diverses llibreries. Això ha permès a aquest treball explicar la teoria de les xarxes neuronals sobre un exemple real. També s'ha vist com diverses llibreries basades amb Python permeten facilitar la tasca de programació en el cas pràctic.

S'ha aconseguit desenvolupar un programa capaç de llegir informació provinent d'un CRM i efectuar previsions amb un gran grau d'encert sobre si les diverses oportunitats s'acabarien convertint en ventes o no.

Tot aquest desenvolupament s'ha realitzat completament online en un Colab Notebook cosa que permet escalar aquest procés. S'ha entès com funciona aquesta eina, les seves limitacions i el seu potencial.

Finalment s'han exposat diverses suggereixes de futures millores que podrien convertir el software desenvolupat en un producte que es pogués comercialitzar.

10. Agraïments

Cal agrair al tutor d'aquest treball, Lluís Solano , la seva guia i ajuda al estructurar les diferents etapes del treball així com les seves ensenyances prèvies sobre Python a la carrera que han estat fonamentals per el desenvolupament d'aquest treball.

Apart d'ell la majoria d'ajuda concreta s'ha obtingut online. Especialment en la documentació de TensorFlow i el Curs de Coursera sobre deep learning de Andrew. Ng. S'agraeix molt que experts en la matèria dediquin part del seu temps a preparar material i ensenyar persones que s'inicien en aquest món.

11. Bibliografia

Referències bibliogràfiques

[1] Google – Google Colab. Visitat 15 Abril 2018, https://colab.research.google.com/notebooks/welcome.ipynb#recent=true

[2] Tensorflow – Tensorflow. Visitat 25 Abril 2018, https://www.tensorflow.org/

[3] Tensorflow – TensorFlow Eager Execution. Visitat 25 Abril 2018, https://www.tensorflow.org/guide/eager

[4] Coursera – Deep Learning Specialization. Visitat 20 Febrer 2018, https://es.coursera.org/specializations/deep-learning

[5] Tensorflow – TensorFlow Tutorials.Visitat 20 Febrer 2018, https://www.tensorflow.org/tutorials/

[6] IBM – Watson. Visitat 20 Juliol 2018, https://www.ibm.com/watson/

[7] X.ai - X . Visitat 7 Abril 2018, https://x.ai/

[8] Crystal - The world's largest personality platform | Free personality test. Visitat 8 Abril 2018, https://www.crystalknows.com/

[9] Tamr Inc - Tamr . Visitat 8 Abril 2018, https://www.tamr.com/

[10] Conversica - AI software for marketing & sales. Visitat 9 Abril 2018, https://www.conversica.com/

[11] Recorded Future - Threat Intelligence Powered by Machine Learning. Visitat 9 Abril 2018, https://www.recordedfuture.com/

[12] Data Word – CRM data set. Visitat 25 Abril 2018, https://data.world/sparklesquad/crm-dataset

[13] TechMergence – CRM Artificial intelligence. VIsitat 20 Agost 2018, https://www.techemergence.com/crm-artificial-intelligence-trends-across-salesforce-oracle-sap/

12. Annexos